

General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

NASA TM-73869

(NASA-TM-73869) AN AUTOMATED PROCEDURE FOR
CALCULATING SYSTEM MATRICES FROM
PERTURBATION DATA GENERATED BY AN EAI PACER
AND 100 HYBRID COMPUTER SYSTEM (NASA) 44 p
HC A03/MF A01 CSCI 00

Unclas
57801

by Edward J. Milner and Susan M. Krosel
Lewis Research Center
Cleveland, Ohio 44135
December 1977



1. Report No. NASA TM-73869	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle AN AUTOMATED PROCEDURE FOR CALCULATING SYSTEM MATRICES FROM PERTURBATION DATA GENERATED BY AN EAI PACER 100 HYBRID COMPUTER SYSTEM		5. Report Date	
		6. Performing Organization Code	
7. Author(s) Edward J. Milner and Susan M. Krosel		8. Performing Organization Report No. E-9465	
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135		10. Work Unit No.	
		11. Contract or Grant No.	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D. C. 20546		13. Type of Report and Period Covered Technical Memorandum	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract <p>Techniques are presented for determining the elements of the A, B, C, and D state variable matrices for systems simulated on an EAI Pacer 100 hybrid computer. An automated procedure systematically generates disturbance data necessary to linearize the simulation model and stores these data on a floppy disk. A separate digital program verifies this data, calculates the elements of the system matrices, and prints these matrices appropriately labeled. The partial derivatives forming the elements of the state variable matrices are approximated by finite difference calculations.</p>			
17. Key Words (Suggested by Author(s)) Hybrid simulation State variables Linear modeling		18. Distribution Statement Unclassified - unlimited STAR Category 61	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages	22. Price*

AN AUTOMATED PROCEDURE FOR CALCULATING SYSTEM MATRICES FROM
PERTURBATION DATA GENERATED BY AN EAI PACER 100 HYBRID
COMPUTER SYSTEM

by Edward J. Milner and Susan M. Krosel

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

Most control system design techniques assume having available a linear representation of the system to be controlled. Although obtaining a linearized model of a complex, non-linear system is usually tedious and time consuming, this task can be made less so by using the automated procedure described in this report.

Techniques are presented for determining the elements of the A, B, C, and D state variable matrices for systems simulated on an EAI Pacer 100 hybrid computer system. The techniques are generalized to allow use with many different simulation models. An automated procedure systematically generates disturbance data necessary to linearize the simulation model and stores these data on an auxiliary TEKTRONIX model 4922 floppy disk.

A separate digital program verifies the floppy disk data, allows for data modification, if necessary, calculates approximations to the partial derivatives forming the elements of the system matrices, and prints the matrices appropriately labeled.

Use of the floppy disk as a storage medium permits the data generation process on the hybrid computer to be separate from the matrix element calculation process.

INTRODUCTION

The intricate nature of many dynamic systems necessarily requires rather complex simulation models to represent performance over their entire operating range. One is likely to find the model to include a combination of linear and nonlinear algebraic and differential equations and maps. Although such a model is useful for predicting system performance over a wide range of operating conditions, generally, its nonlinear characteristics hinder it from being well-suited for control analyses. A basic assumption of most control design techniques is having available a linear representation of the system to be controlled.

E-9465

By considering operation in the vicinity of a particular operating point, state variable techniques can be employed to obtain a linear approximation of the nonlinear system. Doing so, the system equations can be written in vector notation as:

$$\dot{x} = Ax + Bu \quad (1)$$

$$y = Cx + Du \quad (2)$$

where x is a vector of the system time-varying states, \dot{x} is a vector of the time rate of change of the system states, u is a vector of system inputs, and y is a vector of system outputs. Matrices A , B , C , and D are characterized by:

$$a_{ij} = \frac{\partial \dot{x}_i}{\partial x_j} \quad (3)$$

$$b_{ij} = \frac{\partial \dot{x}_i}{\partial u_j} \quad (4)$$

$$c_{ij} = \frac{\partial y_i}{\partial x_j} \quad (5)$$

$$d_{ij} = \frac{\partial y_i}{\partial u_j} \quad (6)$$

One linearization technique is to obtain finite difference approximations for the partial derivatives which comprise the elements of the system matrices. To generate the partial derivatives, the steady-state level of each state variable and system input is independently stepped in both a positive and negative direction while holding the other state variables and system inputs fixed. Corresponding values of the state variable derivatives and system outputs are recorded for each step change. By calculating the deviation of each state variable derivative and output from its steady-state value and by forming appropriate quotients the matrix elements are formed.

This report describes an automated technique for determining the elements of the A , B , C , and D state variable matrices for systems that are simulated on an EAI Pacer 100 hybrid computer system. The technique is generalized to allow its use with many different simulation models. The data taking process is initiated by manually setting a sense switch at the Pacer 100 control console. The order in which the variables are to be perturbed along with other appropriate data are input from cards. Parameter values are transmitted from

the analog computer to the digital computer by means of analog-to-digital converters (ADC's). Perturbation data generated on the Pacer 100 system are recorded and stored on a TEKTRONIX model 4922 floppy disk. The data are later transferred from the floppy disk to an IBM TSS/360 computer where the matrix calculations are performed. The matrix calculation is contained in the program MATCALC which can generate matrices for systems having an arbitrary maximum of 20 states, 10 inputs, and 20 outputs.

The following sections provide details of the apparatus, the digital program, the perturbation process, the floppy disk data format, and the matrix calculation procedure. An example is provided to illustrate the procedure.

APPARATUS

The data taking technique, described in this section, applies to nonlinear system simulations implemented on an EAI Pacer 100 hybrid computer system. The technique could, of course, be modified for other hybrid computer systems. At the NASA Lewis Research Center, the Pacer 100 system includes the Pacer digital computer with 32K of core, a TEKTRONIX model 4010 terminal with auxiliary TEKTRONIX model 4922 floppy disk unit, two model 681 analog computers, and an interface to allow communication between the computers. That interface consists of 48 analog-to-digital converters (ADC's) and 24 digital-to-analog converters (DAC's).

Analog Setup Requirements

It is assumed that the integrations associated with the system dynamics are performed by the integrators on one or both of the analog computers. The integrator outputs then correspond to the system state variables. The perturbation program requires that each state variable, state derivative, input variable, and output variable be transmitted from the analog computer to the digital computer through the interface. Hence, each parameter of interest must be connected to an ADC. For the purposes of this report, parameters formulated on the "primary" analog console are transmitted to the digital computer via a block of ADC's beginning with ADC 1 and parameters formulated on the "secondary" analog console are transmitted via a block of ADC's beginning with ADC 33.

Digital Program Description

Appendix A contains FORTRAN listings of the major parts of a typical simulation program which incorporates the necessary perturbation logic.

The program MAIN is designed to control the execution of the major parts of the program. As the execution of each part is

completed, control is returned to MAIN. The analog computer setup and checkout takes place during the execution of subroutine SET. This includes setting and checking analog pots, digitally controlled attenuators (DCA's), and digitally controlled function generators (DCFG's). Subroutine STATIC provides for the reading of all amplifier outputs at any desired operating point. Subroutine LOOP is the heart of the simulation insofar as this is the area where the dynamics of the system being simulated are updated. By continually cycling through component maps and table lookups, current values of the system state derivatives are obtained. In general, LOOP will receive information from the analog computer and output information to the analog computer through the interface.

The perturbation program consists of three primary digital subroutines - DATOUT, SGNFIX, and AVG. The function served by subroutine DATOUT is to set the integrator initial condition DCA's to correspond to the steady-state operating point being considered and to systematically perturb the state variables and the system inputs one at a time in both a positive and negative sense. DCA's are used because they can be set by the digital computer without changing modes on the analog computer. Subroutine SGNFIX is used to guarantee that the positive algebraic sign of each parameter is recorded on floppy disk. For example, consider a variable, x , whose range spans the set of all real numbers. Hence, x may have both positive and negative real values. However, the analog computer representation can be either the variable $(+x)$ or the variable $(-x)$ depending on the sign of the reference voltage used. If $(-x)$ were the analog computer variable, subroutine SGNFIX converts it to $(+x)$. For each perturbation, subroutine AVG scans the parameters of interest forming 100 point averages for each of them. These data are then recorded on floppy disk.

Formatting the output data and recording it on floppy disk is accomplished in subroutines ADDUP and DISK, respectively. The sole purpose of subroutine DISK is to control the recording of the data on floppy disk. To record on floppy disk, the floppy disk unit is plugged into the disk interface card on the TEKTRONIX 4010 CRT terminal. The write statements in the program are the same as if the data were being written on the CRT terminal. Data is then automatically written on the CRT terminal and recorded on the floppy disk. Note that everything written on the CRT terminal is also recorded on the floppy disk.

Illustrative Example

Consider the following system of four uncoupled dynamic systems linked through output relationships:

$$ST81 = \int (INPUT1 - ST81) dt \quad \text{where } |INPUT1| \leq 2 \quad \text{and } |ST81| \leq 2 \quad (7)$$

$$ST82 = \int (INPUT2 - ST82) dt \quad \text{where } |INPUT2| \leq 2.5 \quad \text{and } |ST82| \leq 2.5 \quad (8)$$

$$ST83 = \int (INPUT3 - ST83) dt \quad \text{where } |INPUT3| \leq 5 \quad \text{and } |ST83| \leq 5 \quad (9)$$

$$ST84 = \int (INPUT4 - ST84) dt \quad \text{where } |INPUT4| \leq 4 \quad \text{and } |ST84| \leq 4 \quad (10)$$

An analog diagram representing a typical system such as these is presented in Figure 1. Equations (7) and (8) are to be programmed on analog console 1. DCA 45 is to be used as the initial condition pot associated with ST81 and DCA 41 is to be used to set the perturbation amplitude of INPUT1. Likewise, DCA's 75 and 71 shall be used with ST82 and INPUT2, respectively.

Correspondingly, Eqs. (9) and (10) are to be programmed on analog console 2. However, on this console DCA's 75, 71, 45, and 41 shall be used with ST83, INPUT3, ST84, and INPUT4, respectively.

In all cases a 2 percent perturbation amplitude is deemed sufficient.

The solution of this example problem will be followed through in the discussion which follows.

PROCEDURE

The data taking process is activated by setting sense switch 4 and then exiting from the main system calculation subroutine (called LOOP) by setting sense switch 1. Control is then transferred to DATOUT. Hereafter, DATOUT controls reentry into LOOP. An internal check is made that the analog computer was in the operate mode at the time the data taking process was activated. Referring to Figure 2, a card is then read in (3I3) format giving the number of state variables, the number of system inputs, and the number of system outputs formulated on the primary analog console only. A series of cards punched in (F5.1, I3, S7) format are then read. Each card contains information characterizing one of the parameters of interest. For each state variable, the algebraic sign of the integrator output, the initial condition pot number, and the proportion of perturbation amplitude desired are required. For each system input, its algebraic sign, its perturbation pot number, and the proportion of perturbation amplitude desired are required. For each system output only its algebraic sign (as previously discussed in the explanation of subroutine SGNFIX) is required.

The next card read is a card in (3I3) format giving the number of state variables, the number of system inputs, and the number of system outputs formulated on the secondary analog console. Similarly, another series of cards in (F5.1, I3, S7) format are then read characterizing the parameters of interest on the secondary analog console as was just done for the primary console. Table I contains a listing of the cards used for the example and Figure 2 shows the data deck setup.

The order in which these data cards are read is very important. The parameter order determined by the cards must correspond with the parameter order determined by the analog ADC's used. State variable data is followed by system input data, which, in turn, is followed by system output data. Also, the order in which the parameters appear within each of the first two groups is important insofar as the order in which the cards are read determines the order in which the perturbations will occur. Note that the order of parameter perturbations is in itself arbitrary. Limitations on this order are imposed only by the analog console on which the integrator for a state variable or system input is located. As this procedure is defined, all state variables and system inputs on the primary analog console will be perturbed first. Then those on the second analog console will be perturbed. Table II gives the order in which the state variables and system inputs of the example are perturbed. Also, given is the analog console for each parameter and its associated ADC channel.

Following the reading of these data cards, the digital program proceeds to monitor the steady-state integrator outputs and sets DCA's corresponding to the integrator initial condition values. After the steady-state operating point is established, the analog computer is automatically put in IC mode where it remains during the perturbation and data taking process. The data taking process begins with a baseline reading of the steady-state values of the parameters of interest and the recording of them on floppy disk. The set of baseline data is designated as reading 1. Following the taking of this baseline data, each state variable and system input is stepped sequentially in both a positive and negative sense. Table III identifies the perturbations associated with each reading for the example.

For each perturbation, the data written on floppy disk are averages of the parameters of interest. The ADC's are read sequentially 100 times with the corresponding values summed and averaged. Since a parameter may be represented on the analog computer with either positive or negative algebraic sign, each average is multiplied by either (+1) or (-1), depending on the parameter sign. Thus, the data recorded on floppy disk corresponds to the positive algebraic sign of each parameter.

Data associated with each perturbation is recorded on floppy disk and each set of data on floppy disk is assigned its own identifying

reading number. Reading numbers are incremented as each set of data is recorded on floppy disk. Data is recorded on floppy disk according to the following format. The reading number is recorded twice in (2I5) format. The reason for this redundancy is so that it may be used as a check when reading the floppy disk. The parameter averages are recorded on floppy disk as integers (with 32767₁₀ representing full scale on the 32K Pacer) at the rate of 18 values per line together with a line checksum according to the format (18I6,I7). The checksum value is the sum of all the integers output on that particular line. If the number of averages exceeds 18 but is not evenly divisible by 18, the second last line of output will be the *n* remaining averages recorded in (nI6) format and the final line will be the corresponding checksum in (I7) format. If the number of averages is less than 18, the *n* averages will be recorded in (nI6) format with a second line containing the checksum in (I7) format. An example of the floppy disk data format is given in Table IV for the test case.

The final characters required on each line are a "line feed" and "X-off". This ending sequence is necessary to insure proper data transfer to the IBM TSS/360 computer. Because this is not the normal line termination, the contents of the digital computer core location controlling line termination must be changed. The address of this core location is obtained by taking the address of .FOR from the memory map (for the listing given in Appendix A, this address is 17234g) and adding 213g. The address displacement value, 213g, depends on the version of the .FOR routine used. Version E01, December 1975 is in use on the Pacer 100 system at Lewis. After loading the program, this memory cell is opened and its contents set to zero by the user.

The entire procedure is repeated for each perturbation. When the data taking process is completed, an appropriate message is output on the line printer. At this point the system may be returned to normal operation by resetting sense switches 1 and 4.

MATRIX CALCULATION TECHNIQUE

This section describes a matrix calculation technique which uses the IBM TSS/360 computer with data input provided by the floppy disk. The matrix calculation is contained in the program MATCALC. This is a generalized version of a program developed at Lewis for matrix generation from data obtained from a hybrid simulation of the F100 engine.

The MATCALC program has three basic sections: data input, data verification, and matrix generation. The data input section uses a software checksum to insure that the floppy disk data was correctly transmitted to the IBM TSS/360 computer. After the data has been correctly transmitted, the data input section unscales it and stores this unscaled data in an internal program matrix for use in the other two program sections. The data verification section determines: (1) if only one state or input was perturbed for each reading and (2) if the

perturbation size of each state or input was at least 1% of the state's or input's steady-state value. The matrix generation section calculates the approximations to the partial derivatives forming the elements of the A, B, C, and D matrices and prints out the matrices in their final form. A listing of the MATCALC program is given in Appendix B.

Transfer of data from the floppy disk to the 360 computer is controlled by a procdef called READIN. A listing of this procdef is included in Appendix B and the procedure for use is given in Appendix D. The procdef, READIN creates the vs-dataset, 'DTRK' which contains the floppy disk data. The data on disk is a series of readings where each reading is a list of values of the states, state derivatives, inputs, and outputs. The required format of each reading was discussed in the preceding section. There should be $2n+1$ readings on disk for each set of matrices, where n is equal to the number of states plus the number of inputs. The first reading is the 'base run' which is a reading of the unperturbed, steady-state values of the states, etc. (NOTE: The MATCALC program terminates if there is no 'base run'.) Corresponding to each reading is a disturbance number. The 'base run' must be assigned the disturbance number of 1. Since each state and input is perturbed in both a plus and minus direction, there are two readings associated with each state and input. These readings should be assigned disturbance numbers of m and $m+1$ where m is an even integer in the range 2 to $2n$.

The readings contained in dataset, 'DTRK', are used by the data input section of MATCALC to form an internal program matrix, DATA. The rows of the DATA matrix are the readings in the same order as they appeared in the floppy disk data and in dataset 'DTRK'. Each row contains the unscaled values of the corresponding reading (i.e., the n th row contains the unscaled values in the n th reading). The order of the values is the same for the reading and the row. Table V lists the DATA matrix for the example.

The rows of the DATA matrix are used in the data verification and the matrix generation sections of MATCALC. The INDX array, also formed in the data input section, is used in referencing individual rows of the DATA matrix. Table V also includes the INDX array for the example. This referencing is accomplished by:

$$\text{Row No. of DATA matrix} = \text{INDX (DISTURBANCE NO.)}$$

If the Row No. is zero, the reading data associated with that disturbance number was not contained in the floppy disk data (i.e., the data for a plus or minus perturbation of a particular state or input was not recorded).

The NAME array associates each state or input with two disturbance numbers: one for the plus perturbation and the other for the minus

perturbation. This array contains the names of the state variables and the system inputs in the order in which they are perturbed (see Table II). The data for the NAME array is contained in the input dataset, 'QINFO'. 'QINFO' is a vs-dataset created by the user and contains data describing the contents of the floppy disk and specifying the structure of the A, B, C, and D matrices. The format of 'QINFO' is given in Appendix C and Table VI lists the 'QINFO' dataset for the example.

Appendix D contains a detailed procedure for processing of perturbation data which has been stored on a floppy disk. Instructions for operating the TEKTRONIX 4010 CRT terminal and TEKTRONIX 4922 floppy disk drive unit are included. IBM TSS/360 commands for reading the floppy disk data into the IBM TSS/360 computer and for performing the matrix calculations are provided. Also in Appendix D is the procedure to follow should checksum errors occur.

Table VII shows the resulting matrix listing for the example. As the results show, the simulated system consisted of four uncoupled dynamic systems linked only through the output relationships. The linearized system equations are:

$$\dot{x}_1 = -0.9938 x_1 + 0.9922 u_1 \quad (11)$$

$$\dot{x}_2 = -0.9918 x_2 + 0.9888 u_2 \quad (12)$$

$$\dot{x}_3 = -0.9939 x_3 + 0.9878 u_3 \quad (13)$$

$$\dot{x}_4 = -0.9923 x_4 + 0.9861 u_4 \quad (14)$$

$$y_1 = 1.003 x_1 - 0.001022 x_3 \quad (15)$$

$$y_2 = -0.004666 x_1 + 1.002 x_2 + 0.001022 x_3 + 0.001019 u_3 \quad (16)$$

$$y_3 = 0.999 x_3 - 0.001546 x_4 \quad (17)$$

$$y_4 = 1.002 x_4 \quad (18)$$

Program Considerations

Some modification of the perturbation program would be necessary if one of the state variables or inputs were at a limiting value while perturbation data were being taken. Because of that limit, that parameter could not be perturbed in both a positive and negative sense. Hence, proper logic would have to be included in subroutine DATOUT to restrict the perturbation of that parameter to the direction which would not violate the limiting value. Likewise, if the problem size were such that only one analog console were required, subroutine DATOUT could be easily modified to eliminate parallel operations addressed to the secondary analog console.

Use of a separate digital computer (the IBM TSS/360 computer) allows for the processing of the perturbation data and analysis of the resulting matrices to be done without interfering with use of the hybrid computer system. Calculation of the system matrices could have been done on the Pacer digital computer with the perturbation data stored on the Pacer's associated disk.

The format specified for data output to disk is arbitrary and can be changed if desired. The DISK subroutine of the perturbation program and the data input section of the matrix calculation program should be modified for the desired format. A new checksum error check should be developed to insure correct data transfer from disk.

CONCLUDING REMARKS

The A, B, C, and D state variable matrices used in controls design can be readily determined using the procedure described in this report. Data required to obtain a linear representation of the system to be controlled are systematically gathered using the techniques presented in the text. Disturbances are introduced in the nonlinear system one at a time. The user may determine the level of each disturbance and the sequential order in which each is to be introduced. For each disturbance, corresponding levels of essential system parameters are automatically recorded on floppy disk.

Use of the floppy disk as a storage medium permits the data generation process on the hybrid computer to be separate from the matrix calculation process. This floppy disk data is input to a separate digital program which contains the matrix calculation process. A software checksum is used to insure correct data transmission from the floppy disk. This program also checks the data for proper disturbance levels and correct number of disturbances. Appropriate messages are output if errors are detected, thus allowing data modification, if desired. The partial derivatives forming the elements of the system matrices are approximated by finite difference calculations. The system matrices, appropriately labeled, are printed out by this program.

Use of this procedure on a dynamic system simulated on a hybrid computer generates automatically the data required for linearization of the system and determines the matrices which represent the system.

ORIGINAL PAGE 15
OF POOR QUALITY

12

APPENDIX A

SIMULATION AND PERTURBATION PROGRAMS

Main Program

```
C
C.....LMN = ANALOG COMPUTER MODE INDICATOR
C
C      COMMON/VAODE/LMN
C      COMMON/FDISC/LE, IXOFF
C      LOGICAL SENS4
C      CALL QSHYIN (IERKOR, 681, 681)
C..... TAKE ADDRESS OF FOR ADD ADD 7213. SET CONTENTS TO ZERO.
C      LE = 7212
C      IXOFF = 7223
C      CALL SET
C      CALL STATIC
10 CALL QSC (6, IERKOR)
CALL QSC (1, IERKOR)
CALL LOOP
IF (SENS4) GO TO 30
CALL STATIC
20 IF (LMN EQ 4) CALL QSDP (IERKOR)
GO TO 10
30 CALL DATOU (LMN)
WRITE (16, 61)
61 FORMAT ('0003HRESET SSWCH, THEN RESET SSWCD')
LA LE
OCT 3001
LA IXOFF
OCT 3001
40 IF (SENS4) GO TO 40
GO TO 20
END
```

SUBROUTINE SET

```
C
C      ENTER YOUR POTS, DCBS, AND
C      DCFGs TO BE SET AND CHECKED
C
C      RETURN
END
```

13

```

SUBROUTINE STATIC
COMMON/FLDISC/LF, INOFF
C
C   ENTER YOUR AMPLIFIERS TO BE READ AND CHECKED
C
WRITE (16, 51)
51 FORMAT ('%24HYOU JUST EXECUTED STATIC?')
LA LF
OCT 5001
LA INOFF
OCT 5001
RETURN
END

```

```

SUBROUTINE LOOP
C
C.... LMN = ANALOG COMPUTER MODE INDICATOR
C
COMMON/XMODE/LMN
LOGICAL SENS4
10 CONTINUE
C
C   ENTER SYSTEM DYNAMIC EQUATIONS INCLUDING
C   BIVARIATE MAPS AND TABLE LOOK-UPS
C
IF (.NOT. SENS4(1)) GO TO 10
CALL USC (2, IERROR)
CALL QRAM1 (LMN)
IF (LMN EQ 4) CALL USH (IERROR)
RETURN
END

```

```

SUBROUTINE DATOUT (IMODE)
C
C.... N1S18 = NUMBER OF STATE VARIABLES ON ANALOG CONSOLE 1
C.... N1I18 = NUMBER OF INPUTS ON ANALOG CONSOLE 1
C.... N1O18 = NUMBER OF OUTPUTS ON ANALOG CONSOLE 1
C.... N2S18 = NUMBER OF STATE VARIABLES ON ANALOG CONSOLE 2
C.... N2I18 = NUMBER OF INPUTS ON ANALOG CONSOLE 2
C.... N2O18 = NUMBER OF OUTPUTS ON ANALOG CONSOLE 2

```

```

C
COMMON/MEAN/N1ST8, N1TOT, N1TOTP, AVR61(50), AMP1SN(25),
1 N2ST8, N2TOT, N2TOTP, AVR62(50), AMP2SN(25), 1RDG
COMMON/FLDISC/LF, IXOFF
DIMENSION N1DCH(25), N2DCH(25)
SCALED FRACTION SGAIN1(25), SGAIN2(25), SV1AMP(25), SV2AMP(25),
1 SVALUE
C.... CHECK THAT COMPUTERS WERE IN OPERATE
IF (MODE EQ 4) GO TO 10
WRITE (16, 61)
61 FORMAT ('%62HCOMPUTERS WERE NOT IN OPERATE WHEN SSWCD WAS SET
1 TRY AGAIN /')
LA LF
OCT 3601
LA IXOFF
OCT 3601
RETURN
C.... CONSOLE 1 STATE VARIABLES, SYSTEM INPUTS, AND SYSTEM OUTPUTS
10 READ (6, 62) N1ST8, N1IN, N1OUT
62 FORMAT (31X)
N1TOT = N1ST8 + N1IN + N1OUT
N1TOTP = N1TOT + N1ST8
READ (6, 63) ((AMP1SN(I), N1DCH(I), SGAIN1(I)), I=1, N1TOT)
63 FORMAT (F5.1, 13, S7)
C.... CONSOLE 2 STATE VARIABLES, SYSTEM INPUTS, AND SYSTEM OUTPUTS
READ (6, 62) N2ST8, N2IN, N2OUT
N2TOT = N2ST8 + N2IN + N2OUT
N2TOTP = N2TOT + N2ST8
READ (6, 63) ((AMP2SN(I), N2DCH(I), SGAIN2(I)), I=1, N2TOT)
C.... READ STEADY-STATE VALUES OF INTEGRATORS
CALL QKRAIS (SV1AMP, 1, N1ST8, IERROR)
C.... CHANGE SIGN TO CORRESPOND TO DCH VALUES
DO 20 I=1, N1ST8
SV1AMP(I) = -SV1AMP(I)
20 CONTINUE
CALL QKRAIS (SV2AMP, 35, N2ST8, IERROR)
C.... CHANGE SIGN TO CORRESPOND TO DCH VALUES
DO 30 I=1, N2ST8
SV2AMP(I) = -SV2AMP(I)
30 CONTINUE
C.... SET INTEGRATOR ICS (DCHS) TO STEADY-STATE VALUES
CALL QSC (0, IERROR)
CALL QSC (2, IERROR)
CALL QNDCH (N2DCH, SV2AMP, N2ST8, IERROR)
CALL QSC (0, IERROR)
CALL QSC (1, IERROR)
CALL QNDCH (N1DCH, SV1AMP, N1ST8, IERROR)
C.... PUT COMPUTERS IN IC
CALL QSC (2, IERROR)
CALL QSC (1, IERROR)
NUMPTS = 100

```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```

      IRDG = 1
      CALL AVG (NUMPTS)
C.... INPUT VALUES
      DO 40 I=1,N1IN
      J = 1+N1ST8
      K = 1+2+N1ST8
      SVIAMP(J) = AVRG1(K)+AMP1SN(J)
40  CONTINUE
      DO 50 I=1,N2IN
      J = 1+N2ST8
      K = 1+2+N2ST8
      SV2AMP(J) = AVRG2(K)+AMP2SN(J)
50  CONTINUE
C.... PERTURB CONSOLE 1 VARIABLES
      CALL QSC (6,IERRO1)
      CALL QSC (1,IERRO1)
      N1VAR = N1ST8+N1IN
      DO 60 I=1,N1VAR
C.... PERTURB VARIABLE POSITIVE
      SVALUE = SVIAMP(I)+SGAIN1(I)*SVIAMP(I)
C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N1ST8) SVALUE = SGAIN1(I)*SVIAMP(I)
      CALL QWDCS (NIDCH(1),SVALUE,1,IERRO1)
      CALL QSDLYR (5,IERRO1)
      CALL LOOP
      CALL AVG (NUMPTS)
      CALL QSC (6,IERRO1)
      CALL QSC (1,IERRO1)
C.... PERTURB VARIABLE NEGATIVE
      SVALUE = SVIAMP(I)-SGAIN1(I)*SVIAMP(I)
C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N1ST8) SVALUE = -SGAIN1(I)*SVIAMP(I)
      CALL QWDCS (NIDCH(1),SVALUE,1,IERRO1)
      CALL QSDLYR (5,IERRO1)
      CALL LOOP
      CALL AVG (NUMPTS)
      CALL QSC (6,IERRO1)
      CALL QSC (1,IERRO1)
C.... RESET VARIABLE TO ORIGINAL VALUE
      SVALUE = SVIAMP(I)
C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N1ST8) SVALUE = .05
      CALL QWDCS (NIDCH(1),SVALUE,1,IERRO1)
60  CONTINUE
C.... PERTURB CONSOLE 2 VARIABLES
      CALL QSC (6,IERRO2)
      CALL QSC (2,IERRO2)
      N2VAR = N2ST8+N2IN
      DO 70 I=1,N2VAR
C.... PERTURB VARIABLE POSITIVE
      SVALUE = SV2AMP(I)+SGAIN2(I)*SV2AMP(I)

```



```

C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N2STB) SVALUE = SGAIN2(C1)*SVZAMP(C1)
      CALL QWDCS (N2DCH(C1), SVALUE, 1, IERROR)
      CALL QSDLYR (C1, IERROR)
      CALL LOOP
      CALL AVG (NUMPTS)
      CALL QSC (C6, IERROR)
      CALL QSC (C2, IERROR)
C.... PERTURB VARIABLE NEGATIVE
      SVALUE = SVZAMP(C1)-SGAIN2(C1)*SVZAMP(C1)
C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N2STB) SVALUE = -SGAIN2(C1)*SVZAMP(C1)
      CALL QWDCS (N2DCH(C1), SVALUE, 1, IERROR)
      CALL QSDLYR (C1, IERROR)
      CALL LOOP
      CALL AVG (NUMPTS)
      CALL QSC (C6, IERROR)
      CALL QSC (C2, IERROR)
C.... RESET VARIABLE TO ORIGINAL VALUE
      SVALUE = SVZAMP(C1)
C.... CHECK IF INPUT VARIABLE
      IF (1.GT.N2STB) SVALUE = .05
      CALL QWDCS (N2DCH(C1), SVALUE, 1, IERROR)
70 CONTINUE
      WRITE (16,64)
64 FORMAT (/'%>CONGRATS. PERTURBATION JOB COMPLETED. /')
      LA 1,F
      OCT 3001
      LA 1,OFF
      OCT 3001
      RETURN
      END

```

SUBROUTINE SGNFIX

```

C
C.... N1TOT = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C.... ON ANALOG CONSOLE 1
C.... N2TOT = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C.... ON ANALOG CONSOLE 2
C
      COMMON/MEAN/N1STB, N1TOT, N1TOTF, AVRG1(C6), AMP1SN(C25),
1      N2STB, N2TOT, N2TOTF, AVRG2(C6), AMP2SN(C25), IRDG
C.... CONSOLE 1 VARIABLES
      DO 30 I=1, N1TOT
      IF (1.GT.N1STB) GO TO 10
      AVRG1(C1) = AVRG1(C1)+AMP1SN(C1)
      GO TO 30

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

10 IF (I.GT.(2*N1ST8)) GO TO 20
   J = 1-N1ST8
   AVR01(I) = -AVR01(I)+AMP1SN(I)
   GO TO 30
20 J = 1-N1ST8
   AVR01(I) = AVR01(I)+AMP1SN(I)
30 CONTINUE
C.... CONSOLE 2 VARIABLES
   DO 60 I=1,N2TOTP
   IF (I.GT.N2ST8) GO TO 40
   AVR02(I) = AVR02(I)+AMP2SN(I)
   GO TO 60
40 IF (I.GT.(2*N2ST8)) GO TO 50
   J = 1-N2ST8
   AVR02(I) = -AVR02(I)+AMP2SN(I)
   GO TO 60
50 J = 1-N2ST8
   AVR02(I) = AVR02(I)+AMP2SN(I)
60 CONTINUE
   RETURN
   END

```

SUBROUTINE AVG (NUMPTS)

```

C
C.... NUMPTS = NUMBER OF POINTS TO BE USED PER AVERAGE
C.... N1TOTP = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C.... ON ANALOG CONSOLE 1
C.... N2TOTP = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C.... ON ANALOG CONSOLE 2
C
COMMON/MEAN/N1ST8,N1TOT,N1TOTP,AVR01(50),AMP1SN(25),
1      N2ST8,N2TOT,N2TOTP,AVR02(50),AMP2SN(25),IRDG
DIMENSION SUM1(50),V1AMP(25),SUM2(50),V2AMP(25),TEMP(25)
DO 10 I=1,N1TOTP
SUM1(I) = 0.
10 CONTINUE
DO 20 I=1,N2TOTP
SUM2(I) = 0.
20 CONTINUE
FLTNO = NUMPTS
C.... GENERATE AVERAGES
DO 50 K=1,NUMPTS
C.... CONSOLE 2
CALL ORRADK (TEMP,33,N2TOTP,ERROR)
DO 30 I=1,N2TOTP
SUM2(I) = SUM2(I)+TEMP(I)
30 CONTINUE

```

```

30 CONTINUE
C.... CONSOLE J
CALL GBKBRK (TEMP, J, N1TOTP, IERROR)
DO 40 I=1, N1TOTP
SUM1(I) = SUM1(I)+TEMP(I)
40 CONTINUE
CALL USC (G, IERROR)
CALL USC (L, IERROR)
CALL LOOP
CALL OSDLYR (G, IERROR)
50 CONTINUE
C.... DETERMINE AVERAGES
DO 60 I=1, N1TOTP
AVRG1(I) = SUM1(I)/FLIND
SUM1(I) = 0.
60 CONTINUE
DO 70 I=1, N2TOTP
AVRG2(I) = SUM2(I)/FLIND
SUM2(I) = 0.
70 CONTINUE
CALL SGNFIX
CALL DISK
RETURN
END

```

```

SUBROUTINE DISK
C
C.... THE ONLY WAY THAT THE FLOPPY DISK CAN BE WRITTEN ON IS THROUGH
C.... THIS SUBROUTINE
C
COMMON/MEAN/N1ST6, N1TOT, N1TOTP, AVRG1(50), AMP1SN(25),
1 N2ST6, N2TOT, N2TOTP, AVRG2(50), AMP2SN(25), IRDG
COMMON/FLDISC/LF, IXOFF
DIMENSION IA(100), IA16(100), IA1(100)
WRITE (1, 61) IRDG, IRDG
61 FORMAT (215)
LA LF
OCT 3001
LA IXOFF
OCT 3001
IRDG = IRDG+1
DO 10 J=1, N1TOTP
IA(J) = AVRG1(I)*32767.
10 CONTINUE
DO 20 J=1, N2TOTP
J = 1+N1TOTP

```

```

      IACJ) = AVERAGE(I)*52767.
20 CONTINUE
      ITOT0 = N1TOTP+N2TOTP
      INDEX = ITOT0/18
      IF (INDEX.LT.1) GO TO 60
      DO 50 K=1, INDEX
        ITOT10 = 0
        ITOT1 = 0
        ITSIGN = 1
        DO 30 J=1, 18
          J = 18*(K-1)+1
          IF (I.EQ.1) ISTART = J
          IF (I.EQ.18) IEND = J
          CALL ADDUP (ITOT10, ITOT1, ITSIGN, IACJ), IRI0(J), IRI(J))
30 CONTINUE
        IF (ITOT10.EQ.0) GO TO 40
        WRITE (1,62) ((IRI0(J), IRI(J)), J=ISTART, IEND), ITOT10, ITOT1
62 FORMAT (18(14, 12), 15, 12)
        LA LF
        OCT 3001
        LA IXOFF
        OCT 3001
        GO TO 50
40 ITOT1 = ITSIGN*ITOT1
        WRITE (1,63) ((IRI0(J), IRI(J)), J=ISTART, IEND), ITOT1
63 FORMAT (18(14, 12), 17)
        LA LF
        OCT 3001
        LA IXOFF
        OCT 3001
50 CONTINUE
60 NUMBER = ITOT0-18*INDEX
      IF (NUMBER.EQ.0) RETURN
      ITOT10 = 0
      ITOT1 = 0
      ITSIGN = 1
      DO 70 J=1, NUMBER
        J = 18*INDEX+1
        IF (I.EQ.1) ISTART = J
        IF (I.EQ.NUMBER) IEND = J
        CALL ADDUP (ITOT10, ITOT1, ITSIGN, IACJ), IRI0(J), IRI(J))
70 CONTINUE
      WRITE (1,62) ((IRI0(J), IRI(J)), J=ISTART, IEND)
      LA LF
      OCT 3001
      LA IXOFF
      OCT 3001
      IF (ITOT10.EQ.0) GO TO 80
      WRITE (1,64) ITOT10, ITOT1
64 FORMAT (15, 12)
      LA LF

```


ORIGINAL PAGE IS
OF POOR QUALITY

```

OCT 3601
LA 1XOFF
OCT 3601
RETURN
80 ITOT1 = ITSIGN*ITOT1
WRITE (1,65) ITOT1
65 FORMAT (17)
LA LF
OCT 3601
LA 1XOFF
OCT 3601
RETURN
END

```

SUBROUTINE ADDUP (ITOT16, ITOT1, ITSIGN, IA, IA16, IA1)

```

C
C..... ITOT16 = RUNNING TOTAL//100
C..... ITOT1 = ABS(RUNNING TOTAL - 100*ITOT16)
C..... ITSIGN = SIGN OF RUNNING TOTAL
C..... IA = NEW NUMBER TO BE ADDED TO RUNNING TOTAL
C..... IA16 = IA/100
C..... IA1 = ABS(IA - 100*IA16)
C
IA16 = IA/100
IA1 = ABS(IA - 100*IA16)
IF (IA16.EQ.0) GO TO 36
IF ((ITSIGN*IA).LT.0) GO TO 16
C..... ITOTAL AND IA HAVE THE SAME SIGN
ITOT16 = ITOT16+IA16
ITOT1 = ITOT1+IA1
IF (ITOT1.LT.100) RETURN
ITOT1 = ITOT1-100
ITOT16 = ITOT16+ITSIGN
RETURN
C..... ITOTAL AND IA HAVE THE DIFFERENT SIGNS
10 IF (ABS(ITOT16).GT.ABS(IA16)) GO TO 20
C..... ITOTAL AND IA CAN BE ADDED
ITOTAL = 100*ITOT16+ITSIGN*ITOT1
ITOTAL = ITOTAL+IA
ITSIGN = 1
IF (ITOTAL.LT.0) ITSIGN = -1
ITOT16 = ITOTAL/100
ITOT1 = ABS(ITOTAL-100*ITOT16)
RETURN
C..... MAGNITUDE OF TOTAL IS GREATER THAN MAGNITUDE OF A
20 ITOT16 = ITOT16+IA16

```

```

ITOT1 = ITOT1-181
IF (ITOT1.GE.0) RETURN
ITOT1 = ITOT1+186
ITOT10 = ITOT10-11516N
RETURN
C.... ASSUME A IS ZERO
X0 181 = 0
RETURN
END

```

MEMORY MAP

C+I+G (E00)			C+I+G (E00)			C+I+G (E00)		
NUMERICAL			ALPHABETICAL			REFERENCE		
NAME	* ADDR	* ZFP	NAME	* ADDR	* ZFP	NAME	* ADDR	* ZFP
1ZZROT	00000		ADDUP	06616		+00574ZZTOP	00106	
1ZZTOP	00106		AVG	04460		+00341MMTOP	21644	
1MMROT	01000		DATOUT	01465		+00121COROT	07332	
1MIN	01000		DISK	05534		+005341ZZROT	00000	
SEL	01136	+00034DISC	57777			+00154MMROT	01000	
STATIC	01141	+00041ABS	11637			+00041COTOP	00000	+0016
LOOP	01177	+00061INLOSS	11637			+00061MIN	01000	
DATOUT	01465	+00121S16N	11660			+00000ELDISC	57777	+0015
SONFIX	03342	+00521TOP	01177			+00061MODE	57776	
AVG	04460	+00341MEAN	57313			QSHYIN	07557	+0000
DISK	05534	+00534MSGFLG	21643			L11	11720	+0001
ADDUP	06616	+00570RT	10056			+0062 H11	11725	+0002
QNDOS	07062	+00320ERSE	10055			SEL	01136	+0003
QKRRDS	07150	+00260CNSTR	07721			STATIC	01141	+0004
QKRRDR	07204	+00470CUNNO	07730			QSC	07753	+0005
QKSWB	07257		QKRM1	07310	+00171TOP	01177		+0006
QKSWNB	07265		QKRRDR	07204	+00475ENSW	12151		+0007
QKRW1	07271		QKRRDS	07150	+0026 S11	11764		+0010
QKRM1	07300		QKCI	10045		QSOE	07512	+0011
QKRM1	07310		QKRM1	07300		DATOUT	01465	+0012
QKRM1	07310	+00170KRW1	07271			FRK	21353	+0013
QKTC1	07320		QKRM1	07300		FRK	13002	+0014
QSDLYR	07333	+00420KSWB	07257			QKRM1	07310	+0017
QSKUN	07466		QKSWNB	07265		QSH	07540	+0020
QSEOP	07466		QKTC1	07320		MEAN	57313	
QSSTOP	07472		QSC	07753	+0005 FRT	21454		+0021
QSEIC	07472		QSC1R	07476		FRK	21413	+0022
QSC1R	07476		QSDLYR	07333	+0042 FRK	12571		+0023
QSOE	07512	+00170SEIC	07472			R11	11732	+0024
QSTC	07517	+00330SELCN	07731			FDT	16675	+0025
QSD	07524		QSEOP	07466		QKRRDS	07150	+0026
QSPC	07524		QSH	07540	+0020 L44	10362		+0027

QSKT	07538	QSHVIN	07557	*0000	N44	10612	*0050
QSFP	07538	QSLC	07517	*0053	H44	10367	*0051
QSS1	07534	QSOB	07524	QSDCS		07062	*0052
QSH	07540	*0020050P	07512	*0011051C		07517	*0053
QSFS	07545	QSPC	07524	HV0		04466	*0054
QSSP	07545	QSPF	07530	L22		10621	*0055
QSHVIN	07557	*000005FS	07545	M22		11300	*0056
QONSTB	07721	QSKT	07538	C24		10253	*0057
QOUNNO	07730	QSKUN	07466	M44		10477	*0040
QSELCN	07731	QSSP	07545	H44		10374	*0041
QUPRLM	07746	QSS1	07534	QSDLYK		07553	*0042
CPU	07751	*006305STOP	07472	S44		10445	*0043
M111	07752	QSTDB	10071	SGN1X		00342	*0052
QSC	07753	*00050TRNS	10075	*0064	H22	10645	*0044
QRC1	10045	QUPRLM	07746	H22		11134	*0045
QKRSF	10055	QSDCS	07062	*0032	C12	10222	*0046
QRI	10056	*00625H01	20156	*010300H0K		07504	*0047
QSDR	10071	SC00	12252	*0105	H22	10717	*0050
QTRNS	10075	*00645000	11637	*0061	D22	11450	*0051
C21	10125	*00045ENSW	12161	*0007D1SK		05534	*0053
C12	10222	*0046SE1	01136	*0003	C21	10125	*0054
C24	10253	*0037SEVERE	21233	*0104	D11	12061	*0055
C42	10337	SGN1X	00342	*0052	M11	12017	*0056
L44	10367	*00275TH01C	01141	*0004H0DUP		06616	*0057
H44	10367	*0031SVENIN	20663	*0100151GN		11660	*0060
H44	10374	*0041SVEN01	20632	*010218BS		11637	*0061
S44	10445	*0043SYSEER	20615	*01010H0		10095	*0062
M44	10477	*0040XMODE	57776	QK1M1		07300	
D44	10530	#00801	57312	QK1C1		07320	
N44	10612	*0030H000P	60000	*00160K540		07257	
L22	10621	*0035HMM00	01000	QK1H1		07271	
H22	10645	*0044HMM0P	21644	QK1M1		07300	
H22	10717	*0050H2200	00000	QK540H		07265	
S22	11114	*722TOP	00106	CPU		07553	*0063
N22	11134	*0045H0H1	21571	*0067USOD		07524	
H22	11300	*0036H11	11732	*00240SKT		07530	
D22	11450	*0051H22	10717	*00500551		07534	
N0RM	11602	*0070H03	12147	QSFS		07545	
HBS	11637	*0061H44	10374	*004105SF		07545	
INLOSS	11637	*0061CRS	20156	*010305FC		07524	
SD00	11637	*0061CKM	17067	*0077QSPF		07530	
IS1GN	11660	*0060CKS	17204	*0072Q5KUN		07466	
L11	11720	*0001CPU	07751	*006305STOP		07472	
L33	11720	*0001C12	10222	*004605CLK		07476	
M11	11725	*0002C21	10125	*0054051C		07472	
H03	11725	*0002C24	10253	*0037050P		07466	
H11	11732	*0024C42	10337	QTRNS	10075	*0064	
S11	11764	*0010D11	12061	*00550H01		10045	
N11	12017	*0056D22	11450	*00510SELCN		07731	
D11	12061	*0055D44	10530	QONSTB	07721		
M11	12142	FHR	12571	*00330HRSF	10055		

HCS	12147	FRT	21434	+0021008	RLH	07746
MCS	12154	FCR	15002	+0014000	NNO	07730
SENSW	12161	+0007	FDT	1667	+0025	M111 07752
F10	12232	+0105	FER	17170	+00760	STDR 10073
SC00	12232	+0105	F10	12232	+0105	KTFNT 21474 +0066
FRK	12571	+0023	FL1	17212	+0074	HR0 21571 +0067
FCR	13002	+0014	FOR	17234		G42 10537
FDT	16675	+0025	FRH	20522	+0071	D44 10530
HRN	16717	+0075	FRK	21413	+0022	S22 11134
GRN	17067	+0077	FST	20536		N0KM 11602 +0070
FER	17170	+0076	FRH	21353	+0013	INLOS 11637 +0061
CRS	17204	+0072	HRN	16717	+0075	S000 11637 +0061
FL1	17212	+0074	H11	11725	+0002	N11 12142
FOR	17234		H22	10645	+0044	L03 11720 +0001
CRS	20156	+0103	H33	11725	+0002	H33 11725 +0002
SR01	20156	+0103	H44	10367	+0031	H33 12147
FRH	20522	+0071	L11	11720	+0001	H33 12154
FST	20536		L22	10621	+0035	FRH 20522 +0071
SYSEFR	20615	+0101	L33	11720	+0001	F10 12232 +0105
SVEHOT	20632	+0102	L44	10362	+0027	S000 12232 +0105
SVENIN	20663	+0100	HR1N	01000		CRS 17204 +0072
SEVERE	21233	+0104	M111	07752		HR1 21256 +0073
HR1	21256	+0073	M11	12017	+0056	FL1 17212 +0074
FRK	21353	+0013	M22	11300	+0036	HRN 16717 +0075
FRK	21413	+0022	M33	12154		FRK 17170 +0076
FRT	21434	+0021	N44	10477	+0040	GRN 17067 +0077
KTFNT	21474	+0066	N0KM	11602	+0070	FOR 17234
HRDT	21571	+0067	N11	12142		SVENIN 20663 +0100
MSGFLG	21643		N22	11134	+0045	SYSEFR 20615 +0101
HM10P	21644		N44	10612	+0030	SVEHOT 20632 +0102
FCUBOT	57312		KTFNT	21474	+0066	CRS 20156 +0103
HEHN	57313		S11	11764	+0010	S001 20156 +0103
XRUDE	57776		S22	11134		FST 20536
FLDISC	57777	+0015	S44	10445	+0043	SEVERE 21233 +0104
FCUTOP	60000	+0016	HR1	21256	+0073	MSGFLG 21643
DN						

APPENDIX B

'MATCALC' PROGRAM AND 'READIN' PROCDEF

C***MATRIX GENERATION ROUTINE

```

DIMENSION NAME(30), INDEX(61)
DIMENSION IVAL(10), DATA(61,70)
DIMENSION NDEL(30), NDESV(20), NDEIV(20), NCOL(30)
DIMENSION SF(70), ATST(30)
DIMENSION A(20,20), B(20,10), C(20,20), D(20,10)
DIMENSION HDG(60), ADG(30), NAM2(20), NAM3(20)

```

```
DATA DATA/4270*0.0/  
DATA LAST/0/, PDXX/61*0/  
DATA PTEST/0.01/, NEPDR/0/  
INTEGER VAR,NDG,ADG  
DATA HDG/60*4H /, ADG/30*4H /
```

```

      READ(5,3001) NS, NC, NY
      FORMAT(3I3)
      NWCODS = 2*NS+NC+NY
      NRUNS = 2*(NS+NC)+1
      NTOT = NS+NC
      NSIZE = NS
      NINPUT = NC

```

```

C      READ(3,8002) (NAME(I),I=1,NTOT)
8002  FORMAT (30A4)
      READ(3,8003) (SF(I),I=1,NWORDS)
8003  FORMAT (8(10F9.2/))
      READ(3,8002) (NAME2(I),I=1,NS)
      READ(3,8002) (NAME3(I),I=1,NY)

```

```

C *****
C *****
C *****
C FLOPPY DISK READ IN AND UNSCALING OF DATA

```

```
C  **READ IN START
    I=C
    1 I=I+1
      READ(5,5001,END=2,ERR=2) NA, NB
5001 FORMAT (2I5)
```

```

C  **DISTURBANCE NUMBER CHECK
   IF (NA .EQ. NB) GO TO 3
   WRITE (6,6100) NA,NB
6100 FORMAT (5X,'NA VALUE (' ,I3,') IS NOT EQUAL TO NB VALUE (' ,I3,') ',/,
-
-ASX,'ENTLE CORRECT DISTURBANCE VALUE (I5) FORMAT')
   READ(7,7101) NA
7101 FORMAT (I5)

```

```

      GO TO 3
      2 WRITE (6,7002)
7002 FORMAT (5X,'NO. OF RUNS NOT EQUAL TO EXPECTED NO. OF DISTURBANCES',/, -
      ASX,'ENTER 1 TO STOP, DEFAULT TO CONTINUE'/)
      READ (7,7003) NTEST
7003 FORMAT (I1)
      IF (NTEST.EQ. 1) STOP
      GO TO 500
      3 IF (INDX(NA).EQ. 0) GO TO 4
      WRITE (6,1001) NA,INDX(NA)
1001 FORMAT (5X,'INDX(',I2,') PREVIOUSLY SET TO SEQ',I4,/, -
      * 5X,'DEFAULT TO REPLACE, 1 TO STOP'/)
      READ (7,7003) NTEST
      IF (NTEST.EQ. 1) STOP

C
C  **INLY - CONTENTS ARE SEQUENCE NUMBERS
      4 INDX(NA) = I
      LCNT = 1
      J = 1
      NEND = 18
      5 JA = J+17
      IF (LCNT.EQ. 1 .AND. NEND.GT. NWORDS) GO TO 10
      IF (J.GT. NWORDS) GO TO 10
      READ (5,5002) (IVAL(K),K=1,NEND),ICK
5002 FORMAT (16I6,I7)
      LCNT = LCNT+1
      GO TO 15
      12 JA = NWORDS
      NEND = JA-J+1
      READ (5,5002) (IVAL(K),K=1,NEND)
      READ (5,5003) ICK
5003 FORMAT (I7)
      LCNT = LCNT+2

C
C  **CHECKSUM CHECK
      15 ISUM = 0
      DO 16 K=1,NEND
      16 ISUM = ISUM+IVAL(K)
      IF (ISUM.NE. ICK) GO TO 50
      100 ILAST = ICK
      WRITE (9,5001) NA,NA
      IF (NEND.NE. 18) GO TO 17
      WRITE (9,5002) (IVAL(K),K=1,NEND),ICK
      GO TO 20
      17 WRITE (9,5002) (IVAL(K),K=1,NEND)
      WRITE (9,5003) ICK

C
C  **UNSCALING OF SAMPLED VARIABLES
      20 K = 0
      DO 25 N=J,JA
      K = K+1
      YI = IVAL(K)

```

ORIGINAL PAGE IS
OF POOR QUALITY

26

```
YS = YI/32768.
25 DATA(I,N) = YS*SF(N)
J = JA*1
IF (J .LE. NWORDS) GO TO 5
IF (I .LT. NRUNS) GO TO 1
WRITE (6,6001)
6001 FORMAT (5X,'MATRIX DATA READ IN AND UNSCALING COMPLETE',-
  ASX,'DATASET ON UNIT 9 CONTAINS (CORRECTED IF NEC) FLOPPY DISK DATA')
GO TO 500

C
C **CHECKSUM ERROR ROUTINE
50 WRITE (6,6101) NA, I, LCNT
6101 FORMAT (' CHECKSUM ERROR',3X,' RUN ',I3,' SEQ ',I3,' LINE ',I4)
WRITE (6,6102) ILAST, ICK, ISUM
6102 FORMAT (' CHECKSUMS : PREVIOUS = ',I7,' READ = ',I7,' CALC = ',I7)
WRITE (6,6103) (IVAL(K),K=1,NEND)
6103 FORMAT (' THE VALUES ARE ',/,1X,18I6,/)
200 WRITE (6,6104)
6104 FORMAT (' ENTER CORRECTED LINE')
IF (NEND .EQ. 18) GO TO 6010
READ (7,5002) (IVAL(K),K=1,NEND)
READ (7,5003) ICK
GO TO 6020
6010 READ (7,5002) (IVAL(K),K=1,NEND),ICK
6020 ISUM = 0
DO 55 K=1,NEND
  ISUM = ISUM+IVAL(K)
  IF (ISUM .EQ. ICK) GO TO 100
  WRITE (6,6105) ISUM,ICK
6105 FORMAT (' CHECKSUMS DIFFER, CALC = ',I7,' READ = ',I7)
  WRITE (6,6106)
6106 FORMAT (' ENTER 1 TO RE-ENTER , 2 TO STOP,DEFAULT TO CONTINUE')
  READ (7,7003) NTEST
  NTEST = NTEST+1
  GO TO (100,200,300),NTEST
300 STOP

C
C *****
C 500 CONTINUE

C
C IF DESIRE A LISTING OF DATA MATRIX WHICH NOW CONTAINS UNSCALED DISTURBANCE DATA,
C INSERT WRITE STATEMENTS HERE
C
C *****
C ** *****
C *****
C
C INITIALIZATION FOR DISTURBANCE VERIFICATION
C
NBRUN = INDY(1)
IF (NBRUN .NE. 0) GO TO 944
WRITE (6,6002)
```


ORIGINAL PAGE IS
OF POOR QUALITY

```

6002 FORMAT (5X,'NO BASE DATA,TERMINATE')
      STOP
544  READ(8,8005) (NDEL(K),K=1,NTOT)
      READ(8,8005) (NDRIV(K),K=1,NY)
      READ(8,8005) (NDBSV(K),K=1,NY)
      READ(8,8005) (NCOL(K),K=1,NTOT)
8005 FORMAT(30I3)
C
      DO 510 N=1,NTOT
      ND = NDEL(N)
510  ATEST(N) = SF(ND)/1000.
C
C      *****
C      *****
C      DISTURBANCE CHECK AND VERIFICATION
C
      DO 700 N=1,NTOT
      NN=N-NSIZE
      NS3 = NAME(N)
      DO 600 M=1,2
      NC=0
      NRI = 2*N*M-1
      NI = INDX(NRI)
      IF (NI.EQ. 0) NI = NBRUN
      DO 400 I=1,NTOT
      ND = NDEL(I)
      IF (ABS(DATA(NBRUN,ND)) .LT. ATEST(I)) GO TO 399
      TEST = (DATA(NI,ND)-DATA(NBRUN,ND))/DATA(NBRUN,ND)
      IF (ABS(TEST) .LT. PTEST) GO TO 400
332  IF (I.NE. N) GO TO 333
      NC = I
      GO TO 400
333  NS1 = NAME(I)
      NEROR = NEROR+1
      WRITE (6,201) NI,NRI,NS1,NS3,DATA(NI,ND),DATA(NBRUN,ND)
201  FORMAT (1H,'SEQUENCE ',I2,' INDX ',I2,2H,'A4,' IS PERTURBED IN ADDITION TO ',A4,/,
      'A2X,'VALUE =',G13.5,' BASE VALUE = ',G13.5)
      WRITE (6,203) NI
203  FORMAT (' ENTER 1 TO USE BASE INSTEAD OF SEQ ',I2,' OTHERWISE DEFAULT')
      READ (7,7003) NTEST
      IF (NTEST.NE. 1) GO TO 400
      INDX(NRI) = 0
      WRITE(6,204) NI
204  FORMAT (5X,'BASE IS BEING USED FOR SEQ ',I2,/)
      GO TO 400
399  TEST = DATA(NI,ND)-DATA(NBRUN,ND)
      IF (ABS(TEST) .GT. 2*ATEST(I)) GO TO 332
400  CONTINUE
      IF (NC.EQ. N) GO TO 600
      IF (INDX(NRI).EQ. 0) GO TO 590
      NEROR = NEROR+1
      ND = NDEL(N)
      WRITE (6,202) NI,NRI,NS3,DATA(NI,ND),DATA(NBRUN,ND)

```

28

```

202 FORMAT (1H , 'SEQUENCE ', I2, ', INDX ', I2, ', ', A4, ' DISTURBANCE LOW OR ZERO', /, -
      ABX, 'VALUE = ', G13.5, ' BASE VALUE = ', G13.5)
      GO TO 600
590 WRITE (6,209) NRI, NS3
209 FORMAT (1H , 'INDX ', I2, ', NO DISTURBANCE FOR ', A4, /, -
      A, ' USING BASE RUN')
600 CONTINUE
700 CONTINUE

C
C *****
C *****
C *****
C MATRIX CALCULATION
C
      J=-1
      DO 800 N=1, NTOT
        NCL = NCOL(N)
        NN = N-NSIZE
        NS3 = NAME(NCL)
        NWORD = NDELT(NCL)
        NTWO = 2*NCL
        NROW1 = INDX(NTWO)
        NROW2 = INDX(NTWO+1)
        IF (NROW1 .EQ. 0) NROW1 = NBRUN
        IF (NROW2 .EQ. 0) NROW2 = NBRUN
607 DNOMN = DATA(NROW1, NWORD) - DATA(NROW2, NWORD)
        IF (ABS(DNOMN) .GT. .001) GO TO 609
        IF (NROW1 .EQ. NBRUN) GO TO 608
        NI = NROW1
        WRITE (6,205) NS3, NI
205 FORMAT (1H , A4, ' DISTURBANCE IDENTICAL, SETTING SEQUENCE ', I2, ' TO BASE')
        NROW1 = NBRUN
        GO TO 607
608 WRITE (6,206) NS3
206 FORMAT(1H , 'NO ', A4, ' DISTURBANCES, MATRIX COLUMN ZEROED')
        DNOMN = 1.0E50
C ***CALCULATION OF A & B MATRIX COLUMNS
609 DO 620 K=1, NSIZE
        NDUM = NDRIV(K)
        IF (K .GT. NSIZE) GO TO 610
        A(K,N) = (DATA(NROW1, NDUM) - DATA(NROW2, NDUM)) / DNOMN
        GO TO 620
610 B(K,NN) = (DATA(NROW1, NDUM) - DATA(NROW2, NDUM)) / DNOMN
620 CONTINUE
C ***CALCULATION OF C & D MATRIX COLUMNS
      DO 630 K=1, NSIZE
        NDUM = NOBSV(K)
        IF (K .GT. NSIZE) GO TO 625
        C(K,N) = (DATA(NROW1, NDUM) - DATA(NROW2, NDUM)) / DNOMN
        GO TO 630
625 D(K,NN) = (DATA(NROW1, NDUM) - DATA(NROW2, NDUM)) / DNOMN

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

630 CONTINUE
    J=J+3
    VAR = NCOL(N)
    IF(N .GT. NS+1) GO TO 710
    IF(N .GT. NS) GO TO 705
    HDG(J) = NAME(VAR)
    GO TO 800
705 J= 2
710 ADG(J) = NAME(VAR)
800 CONTINUE

```

C
C
C
C
C

```

*****
PRINTOUT OF MATRICES
10 COLS MAX PER LINE
NC = NINPUT
WRITE (6,6900)
6900 FORMAT (5X,'THE A MATRIX'/)
    IF (NS .LE. 10) GO TO 6950
    NCOL1=NS/2
    NCOL2=NCOL1+1
    WRITE (6,6960) (HDG(L),L=1,30)
6960 FORMAT(' ',9X,30A4)
    DO 6906 I=1,NS
6906 WRITE (6,6901) NAM2(I), (A(I,J),J=1,NCOL1)
    WRITE (6,6960) (HDG(L),L=31,60)
    DO 6907 I=1,NS
6907 WRITE (6,6901) NAM2(I), (A(I,J),J=NCOL2,NS)
    GO TO 6955
6950 WRITE (6,6960) (HDG(L),L=1,30)
    DO 6908 I=1,NS
6908 WRITE (6,6901) NAM2(I), (A(I,J),J=1,NS)
6901 FORMAT (' ',A4,5X,10G12.4/)
6955 WRITE (6,6902)
6902 FORMAT (5X,'THE B MATRIX'/)
    WRITE (6,6960) (ADG(L),L=1,30)
    DO 6911 I=1,NS
6911 WRITE (6,6901) NAM2(I), (B(I,J),J=1,NC)
    WRITE (6,6903)
6903 FORMAT (5X,'THE C MATRIX'/)
    IF (NS .LE. 10) GO TO 6970
    WRITE (6,6960) (HDG(L),L=1,30)
    DO 6921 I=1,NY
6921 WRITE (6,6901) NAM3(I), (C(I,J),J=1,NCOL1)
    WRITE (6,6960) (HDG(L),L=31,60)
    DO 6922 I=1,NY
6922 WRITE (6,6901) NAM3(I), (C(I,J),J=NCOL2,NS)
    GO TO 6975
6970 WRITE (6,6960) (HDG(L),L=1,30)
    DO 6923 I=1,NY
6923 WRITE (6,6901) NAM3(I), (C(I,J),J=1,NS)
6975 WRITE (6,6904)

```



```
6904 FORMAT(5X,'THE D MATRIX' /)
      WRITE (6,6960) (ADG(L),L=1,30)
      DO 6931 I=1,NY
6931 WRITE(6,6901) NAM3(I),(D(I,J),J=1,NC)
C
C      IF DESIRE A PERMANENT DATASET WITH A,B,C,D MATRICES
C      INSERT WRITE STMTS HERE WITH DESIRED FORMAT
C
C      **ERROR MESSAGE NUMBER PRINTOUT
C      IF (NEROR .NE. 0) WRITE(6,711) NEROR
711 FORMAT (///,' *** AT LEAST ',I3,' ERRORS ***')
      STOP
      END
```

PROCDEF READIN

```
PROCDEF READIN
PARAM DTRK
IF ('DTRK'=''); GO TO 'END'
DISPLAY 'ENTER DATA - FINISH WITH _END'
ERASE DTRK
EDIT DTRK
DEFAULT SYSINX=E
REDIT DTRK
VSEIF DTRK;0
DEFAULT SYSINX=C
DISPLAY 'VS-DATASET COMPLETED - MODIFY IF NEC'
'END'
```

APPENDIX C

'QINFO' dataset

Format:

line 1	NS, NC, NY	3I3
	NAME array	30A4
	SF array	8(10F8.2)
	NAM2 array	30A4
	NAM3 array	30A4
	NDELT array	30I3
	NDRIV array	30I3
	NOBSV array	30I3
last line	NCOL array	30I3

Description:

NS : number of states

NC : number of inputs

NY : number of outputs

NAME: names of the states and inputs in the order perturbed for generation of hybrid data.

length of each name - 4 alphanumeric characters

Each position of the NAME array contains the name of a state or an input. Since each state and each input is perturbed in both a plus and minus direction, each position of the NAME array corresponds to two readings and hence, two disturbance numbers. The correspondence between a position in the NAME array and two disturbance numbers is as follows:

N: position in NAME array

(e.g., NAME(N) = ST1 name of state 1)

disturbance number of plus perturbation 2*N

disturbance number of minus perturbation 2*N+1

NCOL: determines columnar structure of the matrices

The integer values represent the positions of the NAME array in the order desired for the columns of matrices.

(e.g., NAME array ST1, ST2, IN1, IN2, ST3, ST4, IN3, IN4
position 1 2 3 4 5 6 7 8

Want columns 1, 2, 3, and 4 of A and C matrices to be with respect to ST1, ST2, ST3, ST4 and want columns 1, 2, 3, and 4 of B and D matrices to be with respect to IN1, IN2, IN3, IN4.

Then NCOL array is 1, 2, 5, 6, 3, 4, 7, 8)

The data for the arrays SF, NDEL, NDRIV, and NOBSV depend on the order of the values in the rows of the DATA matrix. Let 'ROW' mean the general form of a row in the DATA matrix.

- SF scale factors corresponding to values in 'ROW'.
 the real x values in $1/x$ scale factor representation.
- NDEL: positions in 'ROW' occupied by the values of the states and
 inputs.
- NDRIV: positions in 'ROW' occupied by the values of the state derivatives ordered as the rows of A and B matrices are desired.
- NOBSV: positions in 'ROW' occupied by the values of the outputs ordered as rows of C and D matrices are desired.
- NAM2: names of the state derivatives in order specified in NDRIV.
 length per name - 4 alphanumeric characters.
- NAM3: names of the outputs in order specified in NOBSV.
 length per name - 4 alphanumeric characters.

APPENDIX D

DATA PROCESSING PROCEDURE

Note: This procedure is specialized for use with the NASA-Lewis Research Center EAI Pacer 100 computer system.

TEKTRONIX 4010 TERMINAL INITIALIZATION

1. turn terminal ON (power light will come on).
2. turn 360 switch to the ON position.
3. turn SEL switch to the OFF position.
4. set BAUD switch for 1200.
5. set NORM/TP-WR switch for NORM.
6. set LOCAL/LINE switch for LINE.
7. on the phone modem, momentarily depress the CONNECT button.
(the CONNECT light will come on when line is connected).
8. hit 'BREAK' key on terminal.
9. 'logon' to 360 as normally done.
(check TERMINAL ID; it should be posted).

TEKTRONIX 4922 FLOPPY DISK INITIALIZATION

1. turn POWER switch ON.
2. depress RESET button.
3. turn PROMPT MODE switch OFF.
4. select proper disk.
 - a. check disk notebook.
 - b. make sure disk moves freely in envelope.
5. set DISK switch for disk being used.
 - a. A if disk 1.
 - b. B if disk 2.
6. set TERMINAL switch (switch #1) for terminal being used.
 - a. LEFT if left terminal.
 - b. RIGHT if right terminal.
7. set TRANSMISSION switch (switch #2) to the LEFT.
(LEFT setting - data transmission stops at 'XOFF' character).
8. set R-W switch to MANUAL READ.

DATA TRANSFER FROM FLOPPY DISK

1. depress RESET button on disk.
2. set beginning track address in TRACK ADDRESS.
3. depress LA button on disk.
4. type at terminal
 READIN 'DTRK' (CR).
 when the procdef starts, the message:
 'ENTER DATA - FINISH WITH - END'
 will print at the terminal. Data transfer will be automatic;
 the user can determine this by the periodic "noise" the disk
 will make during this transfer.
5. type at terminal:
 END (CR)
 when the end of the data is reached.
 This will end the EDIT dataset created by procdef READIN.
 It may be necessary to type END (CR) more than once before
 the 360 accepts it. These unaccepted lines are entered into
 the dataset 'DTRK' and must be deleted through REDIT
 before using 'DTRK' as input to the MATCALC program.

'MATCALC' PROGRAM EXECUTION

After creation of datasets 'DTRK' and 'QINFO', the program can be executed by typing at the terminal:

```
DDEF  FT05F001,VS, 'DTRK'
DDEF  FT08F001,VS, 'QINFO'
DDEF  FT09F001,VS, 'QOUTPUT'
MATCALC
```

NOTES:

1. When the message 'MATRIX DATA READ IN' etc. is printed at the terminal, depress RESET button on disk.
2. Final printout at terminal will be the A, B, C, and D matrices
3. If errors are detected, appropriate messages will be printed out, allowing the user opportunity to make corrections. The procedure for checksum error correction follows.

CHECKSUM ERROR PROCEDURE

If the read and calculated values of the checksum for a line of floppy disk data are not equal, a checksum error has occurred. An error message will print at the terminal, allowing the data line to be re-entered as follows:

1. set LOCAL/LINE switch to LOCAL on terminal.
2. depress RESET button on disk.
3. set beginning track address in TRACK ADDRESS.
4. depress LA button.
5. at the terminal,
 - the 'CTRLQ' key will input one line of floppy disk data to the terminal CRT screen.
 - a. Lines of data should be inputted until the proper disturbance reading data is reached. This is done by comparing the RUN number outputted in the error message to the values in the line containing the disturbance number for each reading.
 - b. Lines of data from that reading should then be inputted, noting the checksum for each line. When the checksum equal to the PREVIOUS checksum (printed in the error message) is reached, the LOCAL/LINE switch should be set to LINE.
 - c. The next data line should be inputted by the 'CTRLQ' key. This line will be accepted by the 360 as the re-entered data line. If the checksum of this data line is on the next line, that line will automatically input to the 360.

TABLE I. - PERTURBATION CONTROL

CARDS FOR 4-STATE, 4-INPUT,

4-OUTPUT EXAMPLE

2	2	2	
1.	45		.02
1.	75		.02
-1.	41		.02
-1.	71		.02
-1.			
-1.			
2	2	2	
1.	75		.02
1.	45		.02
-1.	71		.02
-1.	41		.02
-1.			
-1.			

TABLE II. - PERTURBATION ORDER OF STATE

VARIABLES AND SYSTEM INPUTS FOR 4-STATE,

4-INPUT, 4-OUTPUT EXAMPLE

Parameter	Console	ADC Channel
ST1	Primary	1
ST2	Primary	2
IN1	Primary	5
IN2	Primary	6
ST3	Secondary	33
ST4	Secondary	34
IN3	Secondary	37
IN4	Secondary	38

Note: The perturbation order of state variables and system inputs per console is arbitrary.

TABLE III. - READING NUMBERS FOR 4-STATE,
4-INPUT, 4-OUTPUT EXAMPLE

Rdg. # (= disturbance #)

1	BASE RUN
2	ST1 +
3	ST1 -
4	ST2 +
5	ST2 -
6	IN1 +
7	IN1 -
8	IN2 +
9	IN2 -
10	ST3 -
11	ST3 -
12	ST4 +
13	ST4 -
14	IN3 +
15	IN3 -
16	IN4 +
17	IN4 -

Order of values per reading:

ST1, ST2, DRV1, DRV2, IN1, IN2, OUT1, OUT2, ST3, ST4, DRV3, DRV4,
IN3, IN4, OUT3, OUT4

Note - STJ represents state variable x_J where $J = 1$ to 4

INJ represents input variable u_J where $J = 1$ to 4

OUTJ represents output variable y_J where $J = 1$ to 4

DRVJ represents state derivative \dot{x}_J where $J = 1$ to 4

38

16142	24500	0	0	16381	24574	16384	24581	24606	16419	0	0	24576	16382	24598	16409
245898															
2	2														
16119	24595	-135	0	16383	24575	16709	24581	24603	16416	0	0	24576	16383	24595	16410
246210															
3	3														
16076	24592	104	0	16383	24575	16064	24584	24603	16416	0	0	24576	16383	24595	16410
245561															
4	4														
16188	25091	0	-508	16383	24575	16385	25080	24603	16416	0	0	24576	16383	24595	16410
246177															
5	5														
16188	24112	0	463	16383	24575	16385	24099	24602	16416	0	0	24576	16383	24595	16410
245367															
6	6														
16388	24596	104	0	16698	24575	16385	24581	24602	16416	0	0	24576	16383	24595	16410
246509															
7	7														
16389	24597	-135	0	16054	24575	16385	24581	24603	16416	0	0	24576	16383	24595	16410
245229															
8	8														
16388	24596	0	463	16383	25085	16385	24581	24602	16416	0	0	24576	16383	24595	16410
246844															
9	9														
16399	24597	0	-507	16383	24084	16385	24581	24603	16416	0	0	24576	16383	24595	16410
246895															
10	10														
16391	24599	0	0	16381	24575	16384	24582	25098	16418	-513	0	24576	16382	25089	16410
246172															
11	11														
16188	24397	0	0	16383	24575	16385	24581	24120	16415	459	0	24576	16383	24112	16410
245384															
12	12														
16389	24597	0	0	16383	24575	16385	24581	24603	16748	0	-363	24577	16383	24595	16743
246196															

TABLE V. - 'INDX' ARRAY AND 'DATA' MATRIX FOR 4-STATE, 4-INPUT, 4-OUTPUT EXAMPLE

INDX ARRAY																
	1															
	2															
	3															
	4															
	5															
	6															
	7															
	8															
	9															
	10															
	11															
	12															
	13															
	14															
	15															
	16															
	17															

DATA MATRIX																
Row	Column															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.00049	1.87083	0.000000	0.000000	0.999817	1.87485	1.00000	1.87538								
2	1.02045	1.87645	-.204468E-010	0.000000	0.999939	1.87492	1.01984	1.87538								
3	0.981201	1.87622	0.185547E-010	0.000000	0.999939	1.87492	0.980469	1.87561								
4	1.00024	1.871429	0.000000	-.387573E-010	0.999939	1.87492	1.00006	1.91345								
5	1.00024	1.83960	0.000000	0.353241E-010	0.999939	1.87492	1.00006	1.83861								
6	1.00024	1.87653	0.185547E-010	0.000000	1.01917	1.87492	1.00006	1.87538								
7	1.00031	1.87660	-.204468E-010	0.000000	0.979858	1.87492	1.00006	1.87538								
8	1.00024	1.87653	0.000000	0.353241E-010	0.999939	1.91231	1.00006	1.87538								
9	1.00031	1.87660	0.000000	-.386810E-010	0.999939	1.83746	1.00006	1.87538								
10	1.00043	1.87675	0.000000	0.000000	0.999817	1.87492	1.00000	1.87546								
11	1.00024	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								
12	1.00031	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								
13	1.00024	1.87653	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								
14	1.00024	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								
15	1.00024	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87531								
16	1.00031	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								
17	1.00024	1.87660	0.000000	0.000000	0.999939	1.87492	1.00006	1.87538								

9	10	11	12	13	14	15	16	
1	3.75458	2.00427	0.000000	0.000000	3.75000	1.99976	3.75275	2.00305
2	3.75412	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
3	3.75412	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
4	3.75412	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
5	3.75397	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
6	3.75397	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
7	3.75412	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
8	3.75397	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
9	3.75412	2.00391	0.000000	0.000000	3.75000	1.99988	3.75290	2.00317
10	3.82965	2.00415	-.782776E-010	0.000000	3.75000	1.99976	3.82828	2.00317
11	3.68042	2.00378	0.700378E-010	0.000000	3.75000	1.99988	3.67920	2.00317
12	3.75412	2.04443	0.000000	-.443115E-01	3.75015	1.99988	3.75290	3.04382
13	3.75412	1.98545	0.000000	0.340576E-01	3.75015	1.99822	3.75305	1.95472
14	3.75412	2.00391	0.698853E-010	0.000000	3.82523	1.99988	3.75290	2.00317
15	3.75412	2.00391	-.779724E-010	0.000000	3.67554	1.99988	3.75290	2.00317
16	3.75412	2.00391	0.000000	0.336914E-01	3.75015	2.03931	3.75305	2.00217
17	3.75412	2.00391	0.000000	-.441895E-01	3.75015	1.96033	3.75305	2.00317

TALLE VI. - 'QINFO' DATASET FOR 4-STAGE, 4-INPUT, 4-OUTPUT EXAMPLE

4 4 4													
ST1	ST2	IN1	IN2	ST3	ST4	IN3	IN4						
2.		2.5		2.		2.5		2.		2.5		5.	4.
5.		4.		5.		4.		5.		4.			
DRV1DRV2DRV3DRV4													
OUT1OUT2OUT3OUT4													
1	2	5	6	9	10	13	14						
3	4	11	12										
7	8	15	16										
1	2	5	6	3	4	7	8						

Note: Appendix C gives the required format specifications for 'QINFO' dataset and defines the arrays in the MATCALC program which are initialized by this data.

ORIGINAL PAGE IS
OF POOR QUALITY

41

TABLE VII. - A, B, C, D MATRIX RESULTS FOR 4-STATE,

4-INPUT, 4-OUTPUT EXAMPLE

THE A MATRIX

	ST1	ST2	ST3	ST4
DRV1	-0.9118	0.0000	0.0000	0.0000
DRV2	0.0000	-0.9518	0.0000	0.0000
DRV3	0.0000	0.0000	-0.9439	0.0000
DRV4	0.0000	0.0000	0.0000	-0.9923

THE B MATRIX

	IN1	IN2	IN3	IN4
DRV1	0.9422	0.0000	0.0000	0.0000
DRV2	0.0000	0.9888	0.0000	0.0000
DRV3	0.0000	0.0000	0.9878	0.0000
DRV4	0.0000	0.0000	0.0000	0.9861

THE C MATRIX

	ST1	ST2	ST3	ST4
OUT1	1.000	0.0000	-0.4090E-03	0.0000
OUT2	-0.5332E-02	1.002	0.5112E-03	0.0000
OUT3	0.0000	0.0000	0.9990	-0.1932E-02
OUT4	0.0000	0.0000	0.0000	1.002

THE D MATRIX

	IN1	IN2	IN3	IN4
OUT1	0.0000	0.0000	0.0000	0.0000
OUT2	0.0000	0.0000	0.5097E-03	0.0000
OUT3	0.0000	0.0000	0.0000	0.0000
OUT4	0.0000	0.0000	0.0000	0.0000

ORIGINAL PAGE IS
OF POOR QUALITY

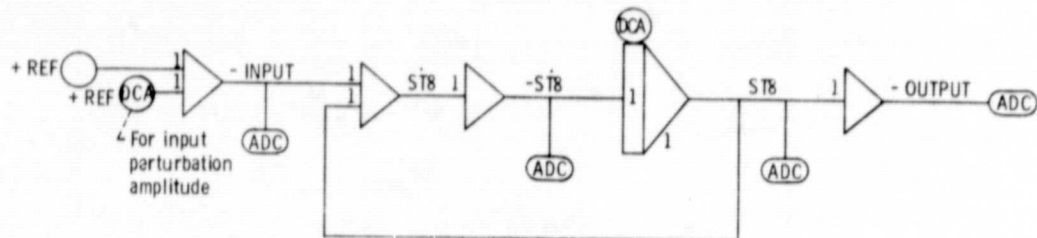


Figure 1. - General analog circuit based on $ST8 = \int (INPUT - ST8)dt$.

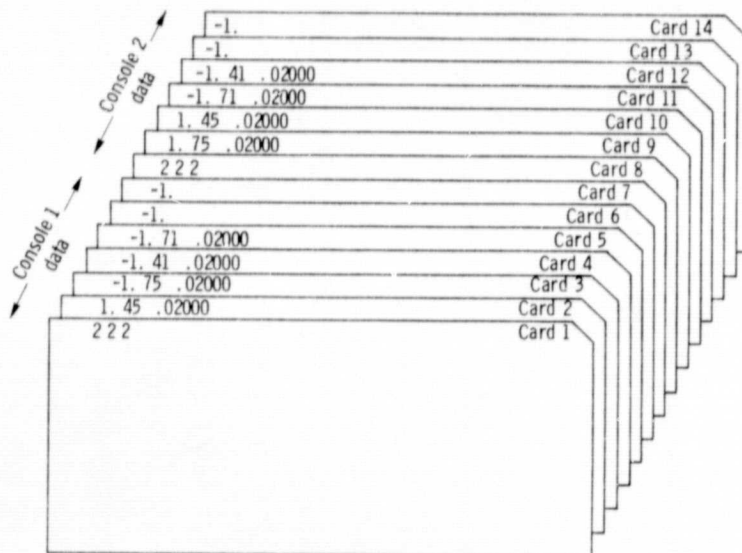


Figure 2. - Data deck setup for example.